PROOF FOUNDATIONS

(W) WEISER DEFINITION OF SLICING:
Given a program P, a slicing criterion C=<v,s> where v is a variable at statement s, and a slice S:
If P halts on input I, then the value of v at statement s each time s is executed in P is the same in P and S. If P fails to terminate normally, s may be executed more times in S than in P, but P and S compute the same values for v each time s is executed by P.

(A) DATA DEPENDENCE:
We say there exists a data dependence between two expressions when the first expression defines the value of a variable and the second one uses this value in at least one of the possible program executions without being any other expression modifying it.
NOTE: We consider that the arguments passed in a function call and the parameters of that function are a specific case of data dependence where the expression changes its name.

(B) CONTROL DEPENDENCE:
There exists a control dependence between two expressions when the second expression cannot be evaluated without evaluating the first expression.

(C) SEQUENTIAL REDUNDANCE:
When the return expression of a block or a function (the last expression of the block in Erlang) is a variable defined in the previous expression, this can be deleted avoiding the definition of this variable and returning the result of the previous expression, taking this expression the last position of the block and being returned in consequence.

(D) SYNTAX ERRROR:
We say there exists a syntax error in a program when the removal or modification of a chosen expression transforms the program into a non-executable state.

(E) SEMANTIC MODIFICATION:
There exists a semantic modification in an expression when the modification of one of its subexpressions modifies the behaviour of the whole expression.

(F) ABSORBING PROPERTY:
A clause of a conditional or a function statement is absorbing when its guard is always evaluated to true or its pattern always matches.

(G) FULL TEST VALIDATION:
There exists full test validation when an original program and a slice extracted from it can be executed with all possible input values of the original program and the values of the slicing criterion are the same in both executions.
NOTE: We consider in this definition also programs with slicing criteria that are independent of program inputs, where there is only one possible execution.

COLOUR LEGEND

Black: Expressions deleted by executing phase 1 (iterative slicing with the selected slicers)
Red: Expressions deleted by executing phase 2 (modified ORBS algorithm)
Green: Expressions remaining in the quasi-minimal slices
Orange: Slicing Criterion

NOTE1: We will not prove whether black expressions of the program code can be deleted or not because they have been deleted by phase 1. Phase 1 produces a complete slice of the original code, so we can guarantee that these expressions are not part of the slice.
NOTE2: Our slices keep the syntax of the original program (we are not interested in amorphous slices). However, in order to make the final slice executable, some modifications of the source code are compulsory (e.g., replacing calls to deleted functions with a constant called "undef"). Therefore, we allow for some modifications of the source code to produce executable slices. The modifications made never affect the behaviour of the source code, they just ensure that the final code is a valid Erlang program.

```
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
%-- bench2.erl
%--
%-- AUTHORS:      Anonymous
%-- DATE:         2016
%-- PUBLISHED:    Software specially developed to test higher order functions and anonymous
%--               functions.
%-- COPYRIGHT:    Bencher: The Program Slicing Benchmark Suite for Erlang
%--               (Universitat Politècnica de València)
%--               http://www.dsic.upv.es/~jsilva/slicing/bencher/
%-- DESCRIPTION
%-- The benchmark receives two input parameters and executes a call to a function using fun
%-- expressions. These expressions are stored in a variable and lately called.
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
```

```
-module(bench2).
-export([main/2]).
main(A,B)->                                 %Given (A), A and B are necessary w.r.t. the if expression
        if                                  %The if expression can be deleted because C is not part of the minimal
                                            slice. After deleting C, there is no other dependence between this
                                            expression and another expression contained in the minimal slice

            A>B ->
                C=f(A,B);
            true ->
                C=f(B,A)
        end,
        C,                                  %C can be deleted because after deleting {C,D} there is no data or control
                                            dependence between this expression and any other expression of the slice.
                                            Variable C is never used
        D = h(fun g/2,A,B),                 %D is necessary because it is the SC
                                            %h(fun g/2,A,B) is the only expression that assigns a value to the SC,
                                            replacing it with undef (NOTE2) would prevent to satisfy (1)&(2)
                                            %Given (A), fun g/2, A and B are necessary w.r.t. h(F,A,B)
                                            %Expressions D and {C,D} are executed after the evaluation of the SC
                                            expression. In consequence, given the lack of a recursive call to the main
                                            function, these expressions cannot modify the value of the SC and can be
                                            deleted
        D,
        {C,D}.

f(X,Y) ->                                   %The calls of the f function have been deleted from the minimal slice, in
                                            consequence this function definition is not part of the slice
        Z= X-Y,
        W= X+Y,
        case Z of
                W ->
                     (fun(B,E) ->
                        N=B*E,
                        B+E
                     end)(X,Y);
                _ -> (fun(N) ->
                        N
                     end)(W)
        end.
g(X,Y) ->                                   %Given (A), X and Y are necessary w.r.t. (fun…)(X,Y)
        (fun(A,B) ->                        %(fun(A,B)->…)(X,Y) cannot be deleted because it is the only expression
                                            of the g function and defines its returned value, in consequence this
                                            expression also defines the value of the SC. Replace it with undef (NOTE2)
                                            would prevent to satisfy (1)&(2)
                                            %Given (A), A and B are necessary w.r.t. A+B
                C=A-B,
                A+B                         %A+B is the only expression of the fun function and cannot be deleted.
                                            Replace it with undef would prevent to satisfy (1)&(2)
                                            %Replace A or B with undef (NOTE2) would prevent to reach the SC due to a
                                            badarith error
        end)(X,Y).                          %Given (A), X and Y are necessary w.r.t fun(A,B)->…
h(F,A,B) ->                                 %Given (A), F, A and B are necessary w.r.t. F(A,B)
        C=B*2,
        D= if
             B>A ->
                B-3;
             B=<A ->
                A+5
           end,
        F(A,B).                             %F(A,B) cannot be deleted because it is the only expression of the h
                                            function and defines its returned value, in consequence this expression
                                            also defines the value of the SC. Replace it with undef (NOTE2) would
                                            prevent to satisfy (1)&(2).
                                            %Given (A), F, A and B are necessary w.r.t. g(X,Y)
```

EXECUTION RESULT:

	SLICING CRITERION
(1) A > B -> A = 2, B = 1 ->	SC = 3 (A+B)
(2) !(A > B) && A =< B -> A = 2, B = 3 ->	SC = 5 (A+B)